VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vývoj aplikací pro iPhone/iPad

iPhone/iPad Application Development

2014

Adam Bardoň

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Adam Bardoň**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vývoj aplikací pro iPhone/iPad
iPhone/iPad Application Development

Zásady pro vypracování:

Zařízení iPad/iPhone patří mezi nejvýznamnější hráče na poli mobilních systémů. Cílem této práce je ilustrovat možnosti systému. Zároveň práce poskytne informace o vývoji na dané platformě a bude se věnovat jejím specifikům, požadavkům, procesům, apod.

1. Popište specifika platformy iOS, a to jak z pohledu vývoje, tak z pohledu distribuce a nasazení aplikace.
2. Navrhněte aplikaci tak, aby byla koncepčně užitečná a bude uživatelsky použitelná. Např. se může jednat o aplikaci využívající přístupné API, která bude vizualizovat fotografie, apod.
3. Implementujte mobilní aplikaci, která bude ilustrovat vybrané klíčové prvky platformy.
4. Pokud to bude možné, zajistěte distribuci aplikace mezi uživatele a tento proces zhodnoťte, včetně případné uživatelské zpětné vazby.

Seznam doporučené odborné literatury:

[1] P. Buttfield-Addison: Learning Cocoa with Objective-C: Developing for the Mac and iOS App Stores, 2012, O'Reilly Media, ISBN: 978-1449318499
[2] J. Conway: iOS Programming: The Big Nerd Ranch Guide, 2013, Big Nerd Ranch Guides, ISBN: 978-0321942050

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 5. května 2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2014 .....

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Michalovi Radeckému, Ph.D., za odbornou a pedagogickou pomoc a za spolupráci. Poděkování patří také mé rodině za podporu během studia.

Abstrakt

Cílem této práce, je seznámit čtenáře s iOS, který patří mezi nejvýznamnější operační systémy pro mobilní zařízení, a rovněž s vývojem aplikací pro tuto platformu. Popíši celý proces - od návrhu, přes implementaci, až po distribuci skrze *App Store*. Práce je rozdělena do čtyř kapitol.

První kapitola obsahuje stručný popis mobilního operačního systému iOS a vývoj pro tuto platformu.

Druhá kapitola se zabývá návrhem mobilní aplikace. Je zde také popsán koncept výsledné mobilní aplikace.

Třetí kapitola ilustruje implementaci vybraných klíčových prvků výsledné mobilní aplikace.

Poslední kapitola pojednává o distribuci aplikace.

Klíčová slova: iOS, iPhone, mobilní aplikace, mobilní zařízení, XCode, Objective-C, App Store

Abstract

The goal of the thesis is to introduce iOS, which belongs among the most significant operating systems for mobile devices, as well as to introduce a development of applications for this platform. I'll describe the whole process - from design, through implementation, to distribution with *App Store*. The thesis is separated to four chapters.

The first chapter contains a brief description of mobile operating system iOS and development for this platform.

The second chapter deals with design of application. The chapter also describes a concept of the final mobile application.

The third chapter illustrates implementation of chosen key elements of the resultant mobile application.

The last chapter discusses about app distribution.

Keywords: iOS, iPhone, mobile apps, mobile devices, XCode, Objective-C, App Store

Seznam použitých zkratek a symbolů

API	–	Application programming interface
ARC	–	Automatic Reference Counting
IAP	–	In-app Purchase
JSON	–	JavaScript Object Notation
PNG	–	Portable Network Graphics
SDK	–	Software development kit
URL	–	Uniform resource locator
WWDC	–	Worldwide Developers Conference

Obsah

1 Úvod **4**

2 Úvod do iOS platformy **6**
2.1 iOS7 6
2.2 Zařízení iOS 7
2.3 Vývoj pro iOS 8
2.4 Objective-C 9

3 Návrh aplikace **13**
3.1 Koncept 13
3.2 Uživatelské rozhraní 14

4 Implementace aplikace **17**
4.1 CocoaPods 17
4.2 Datová komunikace 18
4.3 Lokální ukládání snímků 21
4.4 Animace 23
4.5 Lokalizace 23
4.6 Optimalizace pro 64-bit 25

5 Distribuce **28**

6 Závěr **30**

7 Reference **31**

Přílohy **33**

A Příloha na CD **34**

Seznam obrázků

1 Srovnání grafického rozhraní – nalevo iOS6, napravo iOS7 6
2 Srovnání aplikace Poznámky – nalevo iOS6, napravo iOS7 7
3 Příklad definice bloku 10
4 Prototyp uživatelského rozhraní 15
5 Interface Builder v *XCode* 16
6 Grafické rozhraní Core Data v *XCode* 21
7 ER diagram 22
8 Přidávaní lokalizace v *XCode* 24
9 Lokalizační soubor s již přeloženými řetězci v *XCode* 24
10 *strings* soubor v *XCode* 25
11 obsah *strings* souboru 25
12 Varovná hlášení v *XCode* 26
13 Výsledná aplikace v *App Store* 29

Seznam výpisů zdrojového kódu

1 Definice třídy v Objective-C 9
2 Definice třídy v Javě 10
3 Kategorie SpecialChars 11
4 Inicializace třídy NSBitmapImageRep 12
5 Obsah *Podfile* souboru 17
6 Formát odpovědi 18
7 Metoda pro sestavení URL požadavku 18
8 Metoda pro vykonání požadavku a zpracování odpovědi. 19
9 Data v odpovědi vrácené API metodou GetNewTimeSnaps 20
10 Příklad použití knihovny *SDWebImage* 21
11 Metoda pro vytvoření série 22
12 Příklad použití knihovny Canvas 23
13 Příklad použití NSLocalizedString 25
14 Porovnání datových typů na 32-bitové a 64-bitové architektuře 26
15 Optimalizace metody pro 64-bitové zařízení 27

1 Úvod

Mobilní zařízení se staly nedílnou součástí každodenního života. Stále více přístupů na internet pochází právě z obrazovek chytrých telefonů a tabletů. Od roku 2012 rovněž klesají prodeje počítačů[6]. Tento trend bývá nazýván „Post-PC érou".

Speciálně pro mobilní zařízení se vyvíjí mobilní aplikace, které jsou navržené tak, aby co nejvíce využívaly možností intuitivního uživatelského rozhraní a dotykového ovládání, které mobilní zařízení nabízí. Podle odhadů bylo za rok 2013 globálně staženo 94.4 miliard aplikací[7]. Evropský trh s mobilními aplikacemi tvořící více než 500 miliónů zákazníků představuje velký potenciál pro vývojáře a byznys. Tento sektor zaměstnává zhruba 1 milión vývojářu a 800 tisíc lidi na marketingových a ostatních pozicích. Očekává se, že v roce 2018 by mobilní aplikace mohly zaměstnávat celkově až 4.8 miliónů lidí.

	2012	2013	2014	2015	2016	2017
Bezplatné aplikace	57,331	82,876	127,704	167,054	211,313	253,914
Placené aplikace	6,654	9,186	11,105	12,574	13,488	14,778
Celkem	**63,985**	**102,062**	**138,809**	**179,628**	**224,801**	**268,692**
Bezplatné aplikace %	*89.6*	*91.0*	*92.0*	*93.0*	*94.0*	*94.5*

Tabulka 1: Stažení mobilních aplikací, celosvětově [8]

Z tabulky je zřejmé, že bezplatné aplikace si drží prvenství v počtu stažení. Pro vývojáře jsou IAP a reklamy stále důležitým zdrojem příjmů. V roce 2013 tyto příjmy převyšovaly 6 miliard dolarů[9]. Odhaduje se, že v následujících letech budou příjmy z IAP prudce stoupat a v roce 2017 dosáhnou až 48% všech příjmů. Jedná se tedy o velmi lukrativní byznys, kde lze dosáhnout vysokých zisků, bez nutnosti velkého vstupního kapitálu. Kolem mobilních aplikací v dnešní době vzniká spoustu startupů[1], ke kterým se připojuje mnoho investorů.

Každé mobilní zařízení pohání operační systém. Mezi nejvýznamnější operační systém patří například iOS, Android a Windows Phone. iOS vyvíjí Americká společnost Apple Inc. se sídlem ve městě Cupertino v Kalifornii. Firmu založil v roce 1976 Steve Jobs a Steve Wozniak[10]. Tato firma se specializuje na hardware a software. Jejich nejznámějšími produkty jsou: řada počítačů Mac a MacBook, hudební přehrávač iPod, chytrý telefon iPhone a tablet iPad. Pro své počítače vyvíjí operační systém OS X, jehož aktuální verze vydána 22. října 2013 je 10.9.2. Dále pak vyvíjí např. kancelářský balíček iWork, internetový

[1]Startup (též start up či start-up) je pojem označující nově vznikající projekt či začínající firmu často ještě ve fázi tvorby podnikatelského záměru.[11]

prohlížeč Safari, program pro střih a zpracování videa iMovie nebo program pro tvorbu a úpravu hudby Garage Band.

2 Úvod do iOS platformy

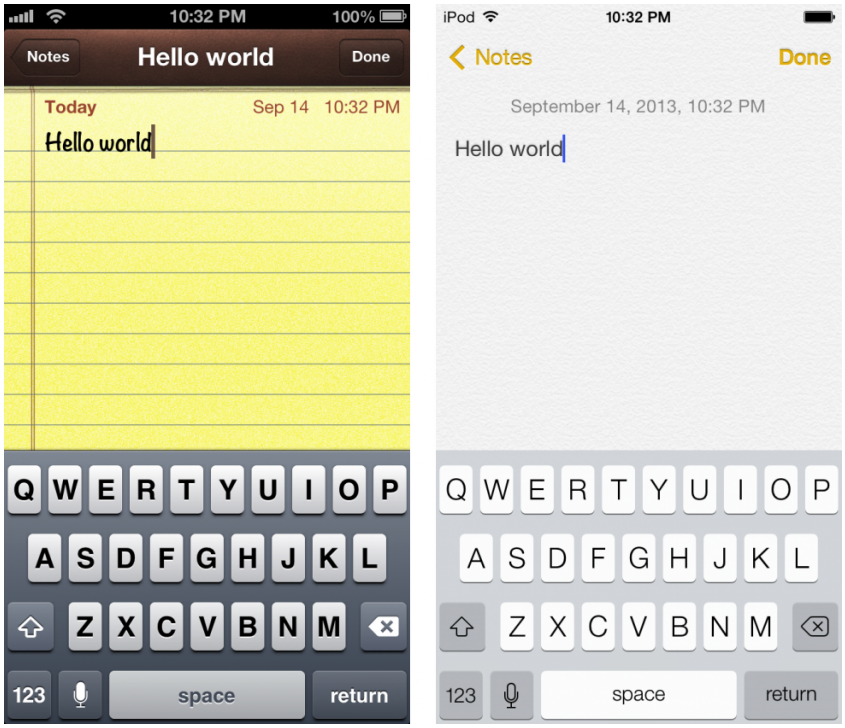
iOS byl poprvé představen v roce 2007 spolu s iPhonem[2]. Jedná se o odlehčenou verzi operačního systému OS X, která ale přidává podporu dotykového ovládání. iOS nabízí neustále se rozrůstající kolekci funkcí, mezi které patří např. iCloud pro zálohu a synchronizaci dat do ostatních zařízení, mapy, FaceTime pro videohovory nebo audiohovory, streamingová hudební služba iTunes Radio[3] a AirPlay pro streamování obsahu ze zařízení do televize, reproduktorů atd. V roce 2008 Apple uvolnil SDK, které třetím stranám umožňuje vyvíjet aplikace pro iOS. O něco později pak Apple spustil *App Store*, ve kterém se současně nachází více než 1 milión aplikaci, a uživateli již bylo staženo více než 60 miliard aplikací[13]. Aktuální verze je 7.1.

2.1 iOS7

Společnost Apple v červnu 2013 na své konferenci WWDC představila v pořadí již sedmou verzi iOS. Ta přinesla mnoho zásadních změn. Tou největší je nepochybně kompletně přepracované grafické rozhraní. Doposud uživatele téměř všude doprovázel skeuomorfismus. Tedy aplikace, které plnily funkci určitého předmětu v reálném světě, imitovaly jeho grafické prvky. Např. aplikace Poznámky napodobovala skutečný zápisník. V iOS7 se však přešlo kompletně na tzv. „flat“ design.

Obrázek 1: Srovnání grafického rozhraní – nalevo iOS6, napravo iOS7

[2]Původní název byl iPhone OS, změna přišla až ve verzi 4.0
[3]Tato služba je zatím dostupná pouze v USA a Austrálii[12].



Obrázek 2: Srovnání aplikace Poznámky – nalevo iOS6, napravo iOS7

Jednou z novinek, které sedmá verze iOS přinesla, je Control Center. Jedná se o menu, ve kterém můžeme ovládat hudbu, kontrolovat WiFi, bluetooth, jas, rychlý přístup k aplikacím, jako kalkulačka či minutka, apod. Velká změna se dotkla rovněž *multitaskingu*[4]. Systém je schopný se naučit, které aplikace používá uživatel nejčastěji, podle toho jim přizpůsobit prioritu a aktualizovat jejich stav na pozadí. Rovněž teď nabízí možnost sledovat miniatury aplikací. Další inovací je AirDrop, který umožňuje odesílaní souborů na jiná Apple zařízení. Tato funkce je dostupná pouze u novějších zařízení (iPhone 5 a výš, iPad 4. generace a výš, atd.). Fotoaparát byl rozšířen o panoramatický režim, čtvercové fotografie či různé efekty. Další příjemnou novinkou je možnost automatické aktualizace aplikací.

2.2 Zařízení iOS

iOS oficiálně není distribuován na jiném než Apple zařízení. Kromě iPhonu jej najdete rovněž na tabletech iPad a iPad mini, iPod Touch a Apple TV. Nově byla také představena

[4]Schopnost operačního systému provádět několik procesů současně.

služba CarPlay, která umožňuje integraci iPhonu do auta. Platforma je tedy poměrně uzavřená, z čehož vyplývá několik výhod:

- Operační systém lze optimalizovat přímo pro konkrétní hardware a jeho běh je tak velmi plynulý a stabilní
- Nízká fragmentace zařízení, což ocení hlavně vývojáři a v konečném důsledku i samotní uživatelé.
- Odpadá zde potřeba testovat vytvářenou aplikaci na všech zařízeních.

V době představení iPhone 4s měly všechny iPhony stejnou velikost displeje, konkrétně 3,5 palce. Uspořádání grafického rozhraní aplikace tedy vypadalo na všech zařízeních stejně. To se změnilo s přichodem iPhone 5, který měl 4palcový displej. U tabletu je situace bezproblémová. iPad má 9,7palcový displej, zatímco iPad mini má displej 7,9palcový. Poměr stran se neliší, a tak se aplikace pro daný displej pouze automaticky přeškáluje.

Na začátku vývoje je třeba se rozhodnout, pro jaké zařízení aplikace bude. Lze vytvořit aplikaci pro iPhone, pro iPad nebo univerzální aplikaci. Aplikace pro iPhone jdou spustit i na iPadu, ale jejich velikost je stejná jako na iPhone a zbytek obrazovky je vyplněn černou barvou. Pokud uživatel chce, může aplikaci zvětšit na celou obrazovku, nicméně zhorší se tím kvalita vykreslení. Aplikace pro iPad jsou v *App Store* viditelné pouze na iPadu. Univerzální aplikace mají definované uživatelské rozhraní jak pro iPhone tak pro iPad.

2.3 Vývoj pro iOS

Základním předpokladem pro vývoj iOS aplikací je mít počítač s operačním systémem Mac OS X. Nemusí to však nutně být počítač vyrobený firmou Apple, jedná se pak o tzv. Hackintosh[5]. Vývojovým prostředím pro iOS a OS X aplikace je *XCode*, který lze zdarma stáhnout na *Mac App Store*. *XCode* obsahuje iOS a OS X SDK a řadu vývojářských nástrojů. Jedním z nich je iOS Simulator, který umožňuje testování aplikací. Jedná se o velmi rychlý nástroj, který je dokonce rychlejší než reálné zařízení, protože využívá zdroje počítače, avšak nelze s ním otestovat vše, např. funkce kamery, atd. K testování na reálném zařízení je třeba mít zaplacený iOS Developer Program, jehož roční poplatek činí $99. Jeho zaplacení rovněž umožní distribuci aplikací skrze *App Store*. iOS Developer Program nabízí také další výhody. Jednou z nich je přístup k beta verzím iOS před tím, než jsou vydány pro veřejnost.

[5] Projekt Hackintosh vznikl poté, co společnost Apple v roce 2005 přešla z procesorů PowerPC na procesory Intel, což umožnilo nainstalovat OS X na počítač postavený na architektůře x86. Hackintosh se stává stále populárnějším, a existuje kolem něj obrovská komunita, dělící se na několik podskupin[16].

2.4 Objective-C

Aplikace pro iOS platformu se vyvíjejí v jazyce Objective-C. Jedná se o objektově orientovaný jazyk, vytvořený v 80. letech 20. století. Byl ovlivněn jazyky C a Smalltalk a dnes je používán zejména firmou Apple v jejich operačních systémech iOS a OS X[**?**]. Tato podkapitola se věnuje vybraným klíčovým vlastnostem Objective-C.

2.4.1 Syntaxe

Objective-C byl implementován jako rozšíření jazyka C, do kterého byl přidán systém zasílání zpráv z jazyka Smalltalk. Syntaxe všech neobjektových operací (např. primitivní proměnné, výrazy, deklarace a volání funkcí) je tedy identická se syntaxí jazyka C, zatímco syntaxe objektově orientovaných operací vychází z jazyka Smalltalk.

Třídy jazyka Objective-C se definují uvnitř direktivy `@interface` a jejich implementace se definuje uvnitř direktivy `@implementation`. Tím se Objective-C liší např. od jazyka Java, kde se jak rozhraní, tak implementace třídy definuje na jednom místě, viz výpisy 1, 2.

```
#import "SuperClass.h"
@interface NewClass : SuperClass {
    int instanceVariable;
}
    -(int)method;
    -(int)methodWithParameter:(int)param;
@end
@implementation NewClass
    -(int)method  {
        return [self methodWithParameter:5];
    }
    -(int)methodWithParameter:param {
        return param*2;
    }
@end
```

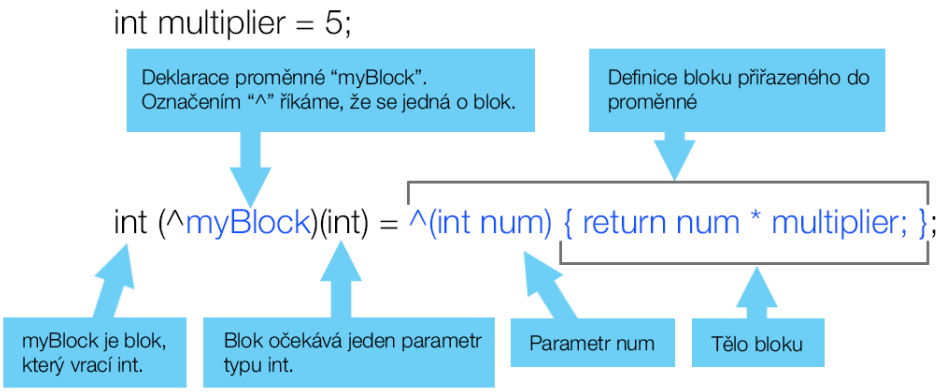
Výpis 1: Definice třídy v Objective-C

```
import com.apress.java.SuperClass;
public class NewClass extends SuperClass
{
    int instanceVariable;
    Integer method() {
        return method(5);
    }
    Integer method(Object param ) {
        return param*2;
    }
}
```

Výpis 2: Definice třídy v Javě

2.4.2 Bloky

Blok je objekt, který umožňuje vytvořit tělo funkce, podobné klasické funkci v C. V jiných jazycích bývá blok někdy nazýván „closure“. Blok lze přiřadit do proměnné a často bývá předáván jako parametr určité metody[**?**].



Obrázek 3: Příklad definice bloku

Jednou z výhod bloku je, že si dokáže zapamatovat stav lexikálního rozsahu platnosti, ve kterém byl definován. Tento stav pak může sdílet a upravovat i v momentě, kdy daný lexikální rozsah platnosti zanikl. V iOS SDK se velmi často používá tzv. *completion block*, např.:

```
[UIView animateWithDuration:0.7
      animations:^{button.alpha = 0.0;}
      completion:^(BOOL finished){ [button removeFromSuperview]; }];
```

Výpis 1: Použití completion bloku

Tento úsek kódu animuje postupné zmizení daného tlačítka. Ve chvíli, kdy je tlačítko kompletně neviditelné, je vykonán blok, který tlačítko odstraní.

2.4.3 Kategorie

Kategorie poskytují schopnost rozšířit funkcionalitu daného objektu, aniž bychom z něj museli dědit, či přímo upravovat jeho implementaci. To oceníme hlavně v případě, kdy chceme přidat metody do již existující třídy, např. `NSString`.

V následující ukázce si ukážeme implementaci kategorie, rozšiřující funkcionalitu třídy `NSString` o metodu, která odstraní speciální znaky ze vstupního stringu.

```
#import "NSString+SpecialChars.h"

@implementation NSString (SpecialChars)

- (NSString *)removeSpecialCharsFromString:(NSString *)string
{
    NSString *stringWithoutSpecialChars = nil;
    NSCharacterSet *specialCharsSet =
   [NSCharacterSet characterSetWithCharactersInString:@"%@#&"];
    stringWithoutSpecialChars = [string  stringByTrimmingCharactersInSet:specialCharsSet];
    return  stringWithoutSpecialChars;
}

@end
```

Výpis 3: Kategorie SpecialChars

Výhodou je, že kategorie mají přístup i k privátním členským proměnným(tzv. property), což je užitečné třeba při použití libovolné knihovny třetí strany. Někdy potřebujeme zjistit stav nějaké property, s čím však autor dané knihovny nepočítal a udělal ji privátní. Přepisovat zdrojový kód knihovny není ideální řešení. Po aktualizaci knihovny na novější verzi bychom naše změny ztratili. Použitím vlastní kategorie tahle starost zaniká.

2.4.4 ARC

ARC je podobně jako garbage collector(Java, C#) metodou pro automatickou správu paměti programu, funguje však zcela odlišně a v iOS se objevil až ve verzi 5. Když se v

Objective-C vytvoří nový objekt, je alokován v paměti a jeho počet referencí je nastaven na 1. Počet referencí je spravován sledováním, kdy si jiné objekty vytvářejí reference na daný objekt a kdy reference zanikají. Pokud je počet referencí daného objektu větší než 0, jeho existence je zaručená. Pokud ale počet referencí dosáhne hodnoty 0, iOS může objekt kdykoliv dealokovat a tím jej zničit.

V minulosti musel programátor kontrolovat počet referencí pomocí speciálních metod(`retain`, `release`, ...). S příchodem ARC tato starost zanikla. Princip ARC je založen na automatickém počítání referencí a provádí se statickou analýzou kódu, tedy již při kompilaci.

Objective-C má dva typy referencí.

- *strong reference* - ARC zvýší počet referencí daného objektu o 1
- *weak reference* - ARC nezvýší počet referencí daného objektu

Typ reference se k dané *property* přiřazuje při její deklaraci.

2.4.5 Jmenná konvence metod

Konvence pojmenovávaní metod v Objective-C je z pohledu jiných programovacích jazyků poněkud netradiční. Před každý argument se dává klíčové slovo, které ma daný argument popisovat. Ve výsledku pak vznikají metody s dlouhým názvem. Dobrým příkladem je metoda pro inicializaci třídy `NSBitmapImageRep`, která vykresluje obrázek z binárních dat a je součástí iOS SDK:

```
- (id)initWithBitmapDataPlanes:(unsigned char **)planes
                    pixelsWide:(NSInteger)width
                    pixelsHigh:(NSInteger)height
                 bitsPerSample:(NSInteger)bps
               samplesPerPixel:(NSInteger)spp
                      hasAlpha:(BOOL)alpha
                      isPlanar:(BOOL)isPlanar
                colorSpaceName:(NSString *)colorSpaceName
                  bitmapFormat:(NSBitmapFormat)bitmapFormat
                   bytesPerRow:(NSInteger)rowBytes
                  bitsPerPixel:(NSInteger)pixelBits;
```

Výpis 4: Inicializace třídy NSBitmapImageRep

3 Návrh aplikace

Proces návrhu aplikace se skládá mimo jiné z konceptu a z návrhu uživatelského rozhraní. V rámci této bakalářské práce byla vypracována aplikace jako migrace Windows Phone aplikace projektu resnap.it. Projekt resnap.it umožňuje vytvářet a sdílet obrázkové galerie s důrazem na časosběrné snímky. Skládá se jak z webové části, představující prostor pro prezentaci a sdílení obrázkových sérií, tak z aplikací, které jsou nástrojem pro vytváření, editaci, správu a prohlížení sérií. Mobilní a desktopová aplikace je dostupná na Windows Store. Mobilní aplikace jsou nabízeny ve dvou režimech. Placená verze aplikace se primárně liší možností vytváření soukromých obrázkových sekvencí a rozsahem možností nahrávání nového obsahu.

3.1 Koncept

V dokumentaci pro vývojáře iOS se píše, že každá aplikace začíná právě jejím konceptem[17]. Nejlepším způsobem, jak definovat koncept aplikace, je určit si problém, který chceme, aby aplikace řešila. V případě aplikace, která je součástí této práce, je otázka konceptu vyřešená. Jelikož se jedná o migraci existující aplikace, funkční specifikace je jasně daná.

Jelikož se jedná o první verzi, aplikace bude prozatím sloužit pouze jako prohlížeč veřejných časosběrných snímků, neboli sérií. Série jsou zobrazeny ve dvou seznamech - nejnovější a nejlépe hodnocené. Po výběru série ze seznamu se zobrazí první snímek, a pokud má série snímku více, lze jimi procházet manuálně nebo spustit prezentaci. Pod snímkem jsou základní informace jako počet snímků, délka trvání série, autor a jiné. Lze rovněž zobrazit podrobnější informace o dané sérii. Uživatel může sérii sdílet, a pokud je přihlášen, může hlasovat nebo nahlásit nevhodný obsah. Série jde také vyhledávat, buď podle klíčových slov nebo štítků.

Aplikace by měla v budoucnu umožňovat vytváření a sdílení sérií. Série můžou být soukromé nebo veřejné. Snímky, které chce uživatel mít pouze pro sebe, nebo je sdílet se svými blízkými (např. pomocí přístupového kódu nebo sociálních sítí), zařadí do soukromých sérií. Veřejné série jsou viditelné i pro ostatní uživatele, a je možné vytvářet také veřejné série s možnosti přispívání dalších uživatelů. Na jedné sérii se tak může podílet široká skupina lidí.

Vytvoření série probíhá v následujících krocích:

- 1. Vyfocení prvního snímku
- 2. Vyplnění údajů jako název série a popis

- 3. Výběr štítků (nepovinné)

Poté je série uložená lokálně v mobilním zařízení, a je na uživateli, zda ji tak ponechá, nebo rovnou nahraje na server. Pro nahrání musí být uživatel zaregistrován a přihlášen. Rovněž je třeba zvolit typ série - soukromá/veřejná. K sérií se lze kdykoliv vrátit, editovat ji a přidávat nové snímky.

Pořizování snímků je primárně v širokoúhlém režimu, pokud tedy uživatel drží mobilní zařízení v orientaci pro portrét, snímek je ořezán na širokoůhly. Oblast snímku, která bude ořezána, by měla být na displeji znázorněna.

V případě, že uživatel přispívá do veřejné série nebo přidává do své série nový snímek, na displeji mobilního zařízení překrývá náhled výsledného snímku předchozí snímek dané série. To se hodí v situacích, kdy uživatel fotí stejný objekt s odstupem času, a může tak lépe přizpůsobit úhel a vzdálenost od objektu. Průhlednost překrytí jde měnit v rozmezí 0 až 100 procent.

Jak již bylo zmíněno, některé funkce jsou přístupné pouze, když je uživatel přihlášen. V aplikaci tedy musí být možnost pro registraci a přihlášení. Aplikace vyžaduje komunikaci se serverovou stranou, která poskytuje API a je postavená na cloudové technologii Microsoft Azure. Aplikace bude jak pro iPhone, tak pro iPad. Podle statistik ze dne 23. března 2014, používá zhruba 85% zařízení iOS7[18]. Starší verze iOS tedy nemá smysl moc řešit, a aplikace budou nabízeny pouze pro iOS7 a výše.

3.2 Uživatelské rozhraní

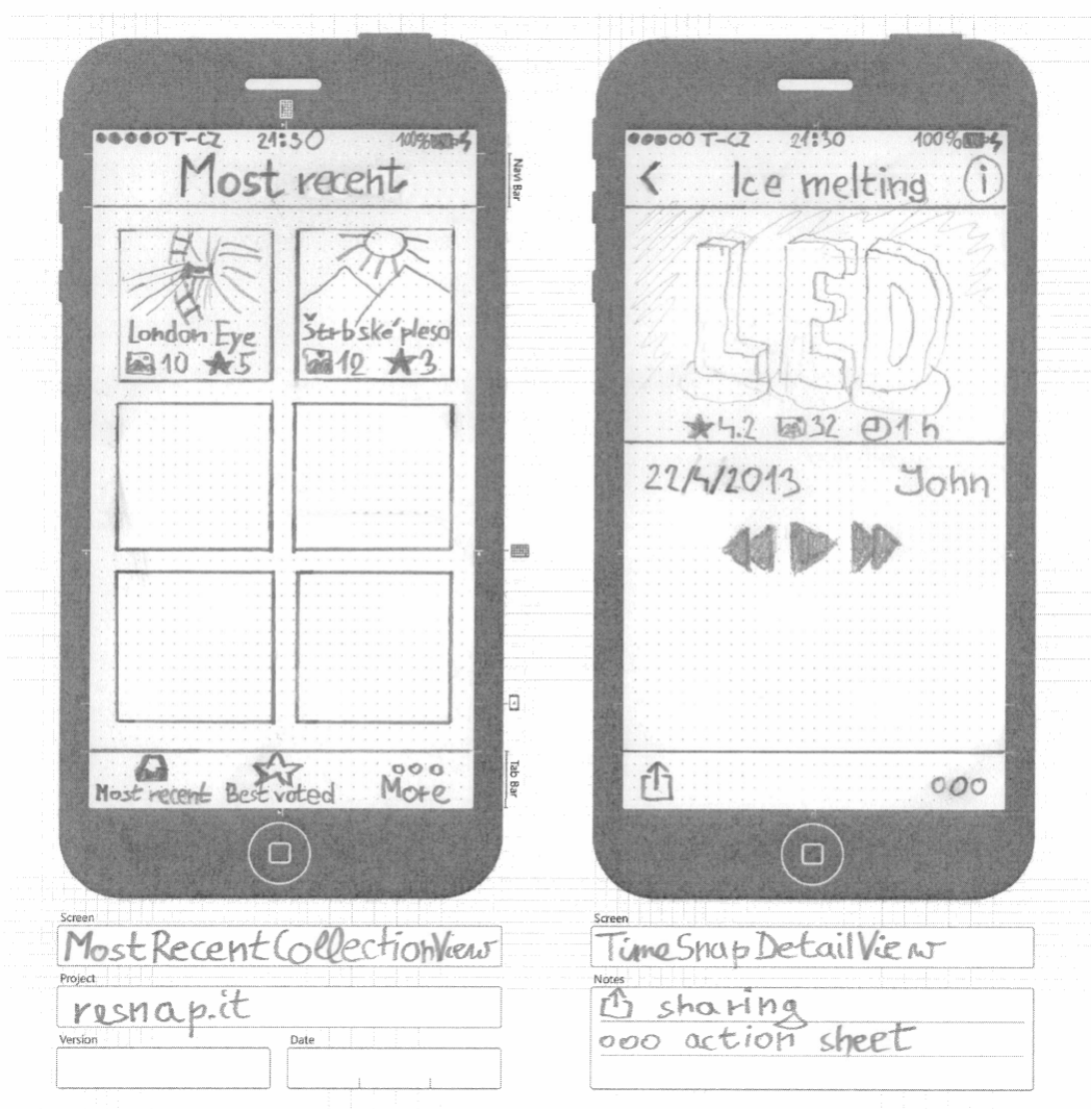
Dalším krokem je návrh uživatelského rozhraní. V této fázi se transformuje koncept do grafické podoby. Design uživatelského rozhraní je velmi důležitým aspektem aplikace. Hraje velkou roli v celkovém dojmu z používání aplikace, a proto by měl být design atraktivní a zároveň praktický.

Uživatelské rozhraní by mělo být konzistentní s operačním systémem, na kterém aplikace běží. Proto je třeba dodržovat pravidla a standardy iOS. Ty shrnuje dokument „Human Interface Guidelines[20]“, který je součástí dokumentace pro vývojáře. iOS ztělesňuje následující motivy:

- UI pomáhá uživateli porozumět a komunikovat s obsahem, nikdy s ním však nezápasí.
- Text je čitelný v každé velikosti, ikony jsou precizní a přehledné, ozdoby jsou přiměřené, důraz je kladen na funkcionalitu.
- Vizuální vrstvy a realisticky pohyb dodávají vitalitu a zesilují uživatelský požitek.

Obsah je tedy na prvním místě a výskyt objektů reálného světa, které s ním nesouvisí, by měl být ojedinělý. Aplikace by měla být přehledná a čitelná. Tomu napomůže kontrast. Dalším důležitým prvkem jsou animace. Animace pomáhají uživateli lépe aplikaci pochopit. Mohou ukazovat odkud určitý prvek UI přišel, nebo naznačit, že určitý prvek je interaktivní. Zároveň pomáhají zlepšit uživatelský zážitek, a zvýšit tak celkový dojem z aplikace.

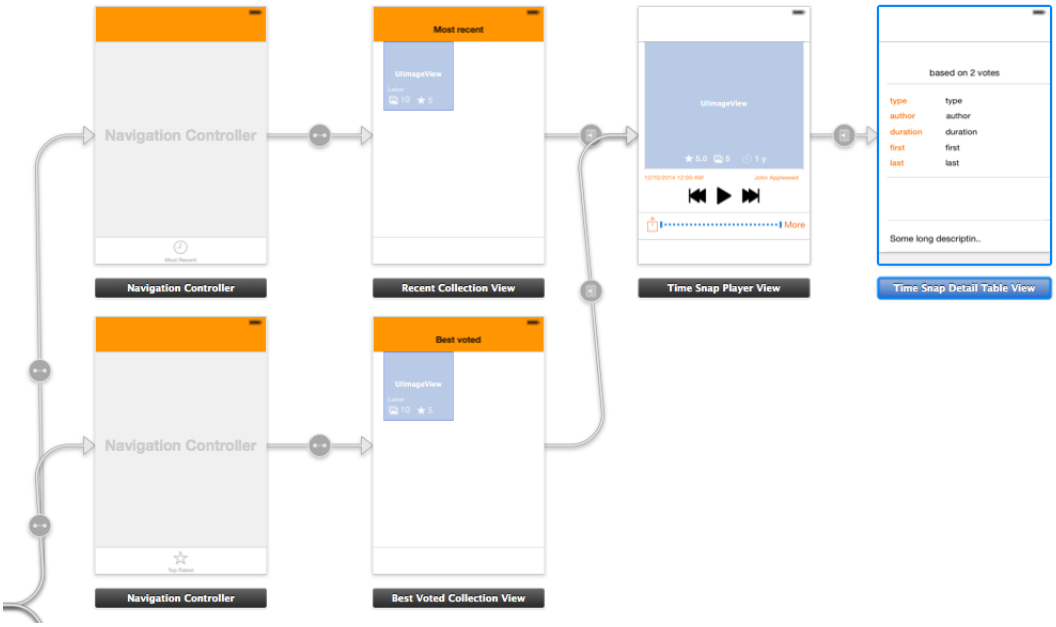
Při návrhu UI je vhodné začít jednoduchou skicou na papíře. Nejsme pak limitování složitostí implementace[17].



Obrázek 4: Prototyp uživatelského rozhraní

3.2.1 *Storyboardy*

Součástí *XCode* je takzvaný *Interface Builder*. Je to nástroj pro tvorbu grafického rozhraní aplikace. S uvedením *XCode* 4.2 a SDK pro iOS5, byla jeho funkcionalita rozšířená o takzvané *Storyboardy*[21]. Storyboardy pomáhají definovat obsah aplikace a přechody mezi jednotlivými obrazovkami. Vše se děje v grafickém prostředí a vývojář nemusí napsat ani řádek kódu. Což ve výsledku znamená kratší čas strávený programováním a méně chyb. V případě univerzální aplikace jsou *Storyboardy* dva - jeden pro iPhone a jeden pro iPad, a ikdyž není problém mít v jednom *Storyboardu* desítky obrazovek aplikace, u komplexnějších aplikací se jich vetšinou vytváří více.



Obrázek 5: Interface Builder v *XCode*

Storyboard se skládá ze scén, kde každá z nich reprezentuje *view controller* a jeho prvky(tlačítka, obrázky, textová pole a mnoho dalších)[22]. Scény jsou propojeny objektem zvaným *segue*, který reprezentuje přechod mezi dvěma *view controllery*. Lze si vybrat ze dvou základních přechodů(*modal*, *push*) nebo naimplementovat vlastní. Mezi scénami lze přenášet data pomocí metody `prepareForSegue:sender:`.

4 Implementace aplikace

4.1 CocoaPods

Při vývoji aplikace resnap.it bylo použito několik knihoven třetích stran. Dobrým nástrojem pro vyhledávání těchto knihoven jsou webové stránky `www.cocoacontrols.com`. Každá položka na těchto stránkách obsahuje odkaz na svůj repozitář na serveru `www.github.com`. Způsobů, jak přidat knihovnu do svého projektu, je mnoho, např. zkopírování zdrojových souborů dané knihovny přímo do složky projektu. To však může v budoucnu způsobit řadu problémů. Elegantnějším řešením je použití nějakého balíčkovacího manažéra, v případě Objective-C(iOS i OS X) existuje tzv. *CocoaPods*[23]. Jedná se o CLI[6] nástroj napsaný v jazyce Ruby. Jeho použití je velmi snadné. Nejdříve je třeba vytvořit v kořenovém adresáři projektu tzv. *Podfile* soubor. Jeho obsah může vypadat třeba takto:

```
platform :ios, '7.0'
pod 'SDWebImage', '~> 3.0', :inhibit_warnings => true
pod 'KASlideShow'
pod 'AXRatingView'
pod 'AMTagListView'
pod 'Canvas'
```

Výpis 5: Obsah *Podfile* souboru

První řádek specifikuje platformu a její verzi. Další řádky pak říkají, které knihovny se mají použít. U každé knihovny může být uvedena verze v jednom z mnoha formátů. Pokud bychom chtěli u dané knihovny vypnout varovná hlášení, použijeme parametr `inhibit_warnings`. Poté stáčí v terminálu přejít do adresářové složky projektu:

```
cd cesta/ke/korenovemu/adresari/projektu
```

A zadat příkaz:

```
pod install
```

Tím se stáhnou nejnovější verze zadaných knihoven. Zároveň to vygeneruje `.xcworkspace` soubor. Od toho momentu se musí pro práci na projektu otevírat `.xcworkspace` soubor místo původního `.xcodeproj` souboru.

Dalším užitečným příkazem je:

```
pod outdated
```

[6] Command-line interface, tedy příkazový řádek

Tento příkaz vypíše seznam použitých knihoven, u kterých je k dispozici nová verze.

Soubor *Podfile* může také odkazovat přímo do nějakého *git* repozitáře, do lokální složky, atd.

4.2 Datová komunikace

Jednou z hlavních činností, které se provádí na pozadí při používání aplikace, je datová komunikace s API serveru `resnap.it`. Komunikace probíhá pomocí HTTP POST požadavků. Server vrací odpovědi ve formátu JSON.

```
{
    "Status":  ["OK" nebo "ERROR"],
    "Data":    [data navracena metodou; v pripade chyby jsou zde informace o chybe]
}
```

Výpis 6: Formát odpovědi

Ihned po načtení aplikace je třeba stáhnout seznam nejnovějších galerií pro jejich zobrazení. Následující metoda nejprve sestaví URL adresu požadavku. Ta obsahuje URL prefix, název metody API a její parametry. Poté je zavolaná metoda, která zašle na server požadavek a zpracuje odpověď. Na konec je vykonán *completion block*, který je předáván na vstupu. Jeho úkolem je inicializace jednotlivých sérií.

```
- (void)getNewTimeSnapsWithCompletion:(void (^)(NSArray* jsonResponse))completion {

    NSString *urlAsString =
      [NSString stringWithFormat:@"%@GetNewTimeSnaps?accessToken=%@", URL_PREFIX,
          ACCESS_TOKEN];
    NSURL *url = [[NSURL alloc] initWithString:urlAsString];

    [self  getJSONWithCompletion:completion fromURL:url forKey:@"Items"];
}
```

Výpis 7: Metoda pro sestavení URL požadavku

Následující metoda vykoná URL požadavek a zpracuje odpověď. Jelikož zde probíhá síťová komunikace, je vhodné zobrazit ve stavovém řádku indikátor síťové aktivity. Toho docílíme pomocí `setNetworkActivityIndicatorVisible:YES`. Za zmínku také stojí `dispatch async(dispatch get main queue(), ^{...}`. Znamená to, že operace uvnitř složených závorek bude vykonána na vedlejším vlákně. To se používá, pokud by určitá operace byla časově náročnější a zablokovala by tak hlavní vlákno, takže by UI nereagovalo na akce uživatele.

```
- (void)getJSONWithCompletion:(void (^)(NSArray* jsonResponse))completion fromURL:(NSURL*)
    url forKey:(NSString*)key {
  NSMutableURLRequest *request = [[NSMutableURLRequest alloc] initWithURL:url];
  [request setHTTPMethod:@"POST"];

  [[UIApplication sharedApplication] setNetworkActivityIndicatorVisible:YES];
  [NSURLConnection sendAsynchronousRequest:request queue:[
      [NSOperationQueue alloc] init] completionHandler:^(NSURLResponse *response, NSData
          *data, NSError *error) {
      if (error) {
          NSString* errorMessage = [NSString stringWithFormat:@"There was an error reading
              new timesnaps feed. %@",
                                    [error localizedDescription]];

          [[AppDelegate instance] showError:errorMessage];
      } else {
          NSError *jsonError;
          NSDictionary *responseJSON = [NSJSONSerialization JSONObjectWithData:data
                                        options:NSJSONReadingMutableLeaves error:&jsonError];
          if (jsonError) {
              NSString* errorMessage = [NSString stringWithFormat:@"There was an error
                  reading new timesnaps feed. %@",
                                        [jsonError localizedDescription]];

              [[AppDelegate instance] showError:errorMessage];
          }
          else {
              NSString *status = [responseJSON valueForKey:@"Status"];
              if ([status isEqual: @"OK"]) {
                  NSDictionary *data = [responseJSON valueForKey:@"Data"];
                  NSArray *items = [data valueForKey:key];
                  dispatch_async(dispatch_get_main_queue(), ^{
                      if (completion){
                          completion(items);
                      }
                  });
              }
          }
      }
  }];
  [[UIApplication sharedApplication] setNetworkActivityIndicatorVisible:NO];
}
```

Výpis 8: Metoda pro vykonání požadavku a zpracování odpovědi.

Ukázka odpovědi serveru:


```
"Items" : [
        {
          "ViewCount" : 277,
          "Code" : "AMjl2JC01",
          "LastSnapAdded" : "/Date(1392910957587)/",
          "Permission" : 2,
          "HasPassword" : false,
          "AzureName" : "1eca3b17d4e...",
          "CoverSnapExtension" : "jpg",
          "LatLngWkt" : null,
          "Name" : "IT4Innovations Supercomputer",
          "AuthorId" : 6,
          "Password" : null,
          "Created" : "/Date(1368646948187)/",
          "VoteSum" : 5,
          "Duration" : 38009760000,
          "Id" : 226,
          "ReportCount" : 0,
          "SnapCount" : 64,
          "Desc" : "The Building of the Super...",
          "Labels" : [
            {
              "LanguageId" : 2,
              "Id" : 11,
              "Name" : "buildings"
            }
          ],
          "Status" : 1,
          "VoteCount" : 1,
          "CoverSnapAzureName" : "f4cc80..."
        },
        ...
]
```


Výpis 9: Data v odpovědi vrácené API metodou GetNewTimeSnaps

Pro samotné stahování snímků galerií je použita knihovna *SDWebImage*[24]. Knihovna poskytuje *kategorii* třídy *UIImageView* pro podporu obrázku umístěných na internetu. Umí asynchronní stahování a ukládání do mezipaměti. Zaručuje, že žádný obrázek nebude stáhnutý z jedné URL vícekrát. Používá ji hodně známých projektů, např. Facebook.

Následující úsek kódu stáhne snímky dané galerie a přidá je do *property* `self.slideshow`, která je instancí třídy *KASlideshow*[25]. *KASlideshow* je další použitou knihovnou v této aplikaci, a jak již název napovídá, jedná se o prezentaci snímku.

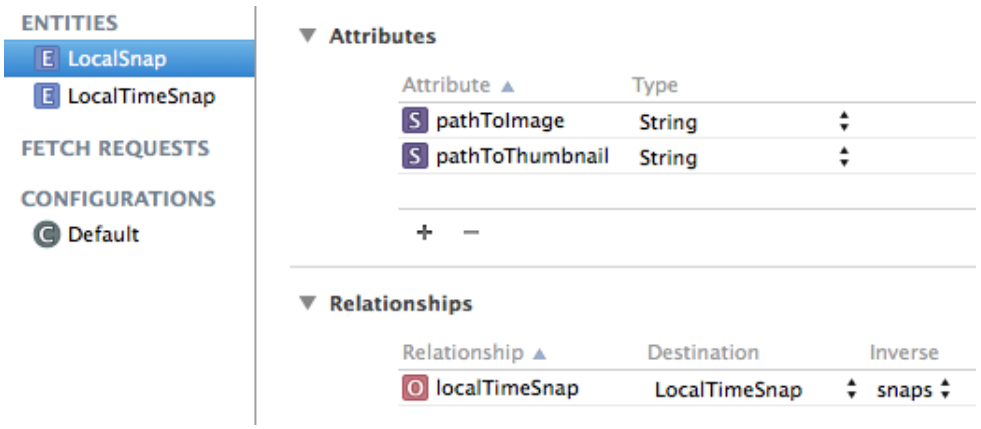
```
SDWebImageManager *manager = [SDWebImageManager sharedManager];
for (NSURL *snap in self.timeSnap.imagesURLArray) {
    [manager downloadWithURL:snap options:0 progress:nil completed:^(UIImage *image, NSError
        *error, SDImageCacheType cacheType, BOOL finished) {
      if (image) {
        [self.slideshow addImage:image];
      }
    }];
}
```

Výpis 10: Příklad použití knihovny *SDWebImage*

4.3 Lokální ukládání snímků

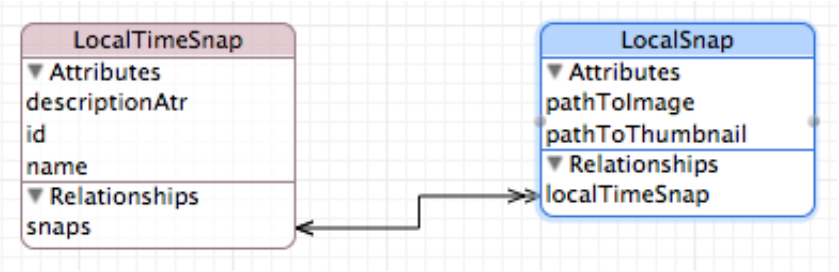
Aplikace umožňuje vytváření lokálních sérií. Jedná se tedy o data, u kterých je třeba zajistit perzistenci. Potřebujeme uložit dva typy dat. Informace o sérii, tj. název, popis, atd. Dále je třeba ukládat samotné snímky, a ke každému snímku také potřebujeme informace, tj. cesta v souborovém systému, id série do které patří, atd. iOS poskytuje rozsáhlou kolekci nástrojů pro ukládání, přístup a sdílení dat. Jedním z nich je tzv. *Core Data*, framework pro ukládání strukturovaných dat a mapování na programové objekty. Nechybí zde ani oblíbený relační databázový systém SQLite.

Pro ukládání informací o sériích a jejich snímcích se v aplikaci používá *Core Data*, který umožňuje definovat datový model aplikace v grafickém rozhraní. Poskytuje infrastrukturu pro operace jako ukládání, načítaní, „zpět“ a „vpřed“.



Obrázek 6: Grafické rozhraní Core Data v *XCode*

Datový model obsahuje dvě entity, viz následující ER diagram:



Obrázek 7: ER diagram

Jak již bylo zmíněno v kapitole 3.1 Koncept, pro vytvoření nové série uživatel vyfotí první snímek, vyplní údaje a zmáčkne tlačítko „Uložit“. Následuje vytvoření nového objektu entity *LocalTimeSnap*. Atribut *id* je typu *date* a je vygenerován podle aktuálního data a času. Dále je vytvořená složka pro snímky dané série, jejíž název je tvořen časovou známkou, tzv. *timestamp*. Poté se vytvoří nový objekt entity *LocalSnap*, obsahující cestu ke snímku a miniatuře snímku v souborovém systému. Na konec je zavolána metoda, která podle vygenerované cesty uloží pořízený snímek a jeho miniaturu na pevný disk zařízení.

```
- (IBAction)saveNewTimeSnap:(id)sender {
    NSManagedObjectContext *context = [self managedObjectContext];

    LocalTimeSnap *newTimeSnap =
        [NSEntityDescription insertNewObjectForEntityForName:@"LocalTimeSnap"
            inManagedObjectContext:context];
    [newTimeSnap setValue:[NSDate date] forKey:@"id"];
    [newTimeSnap setValue:self.nameTextField.text forKey:@"name"];
    [newTimeSnap setValue:self.descriptionTextView.text forKey:@"descriptionAtr"];

    NSString *timestamp = [NSString stringWithFormat:@"%f", [[newTimeSnap valueForKey:@"id"]
        timeIntervalSince1970]];

    LocalSnap *newSnap = [NSEntityDescription insertNewObjectForEntityForName:@"LocalSnap"
        inManagedObjectContext:context];
    newSnap.pathToImage = [NSString stringWithFormat:@"%@/1.jpg", timestamp];
    newSnap.pathToThumbnail = [NSString stringWithFormat:@"%@/1_thumbnail.jpg", timestamp];

    newTimeSnap.snaps = [NSSet setWithObjects:newSnap,nil];

    NSError *error = nil ;
    if (![ context save:&error]) {
```

```
            NSString* errorMessage = [NSString stringWithFormat:@"Can't Save! %@ %@", error,
                [error  localizedDescription ]];
            [[AppDelegate instance] showError:errorMessage];
        }

        [self  saveTakenSnapImageWithName:@"/1.jpg" intoFolder:timestamp
            withThumbnailName:@"/1_thumbnail.jpg"];
        [self  dismissViewControllerAnimated:YES completion:nil];
    }
```

Výpis 11: Metoda pro vytvoření série

4.4 Animace

Pro zjednodušení práce s animacemi prvků UI je v aplikaci použitá knihovna Canvas[26]. Tato knihovna obsahuje 20 předdefinovaných animací. Lze ji použít přímo v *Interface Builderu* nebo pomocí kódu. Stačí si vybrat typ animace a nastavit její délku trvání a zpoždění přehrávání.

```
- (void)animateView:(UIView*)view withDuration:(float)duration withDelay:(float)delay
    withType:(CSAnimationType)type {
    CSAnimationView *animation = [[CSAnimationView alloc]
            initWithFrame:CGRectMake(0, 0, 100, 100)];
    animation.duration = duration;
    animation.delay    = delay;
    animation.type     = type;
    [view.superview addSubview:animation];

    [animation addSubview:view];
    [animation startCanvasAnimation];
}

// prirazeni  animace vybranym prvkum
[self  animateView:self.votesCountLabel withDuration:1 withDelay:0.6
        withType:CSAnimationTypeFadeIn];
[self  animateView:self.descriptionLabel withDuration:1 withDelay:0.15
        withType:CSAnimationTypeSlideUp];
```

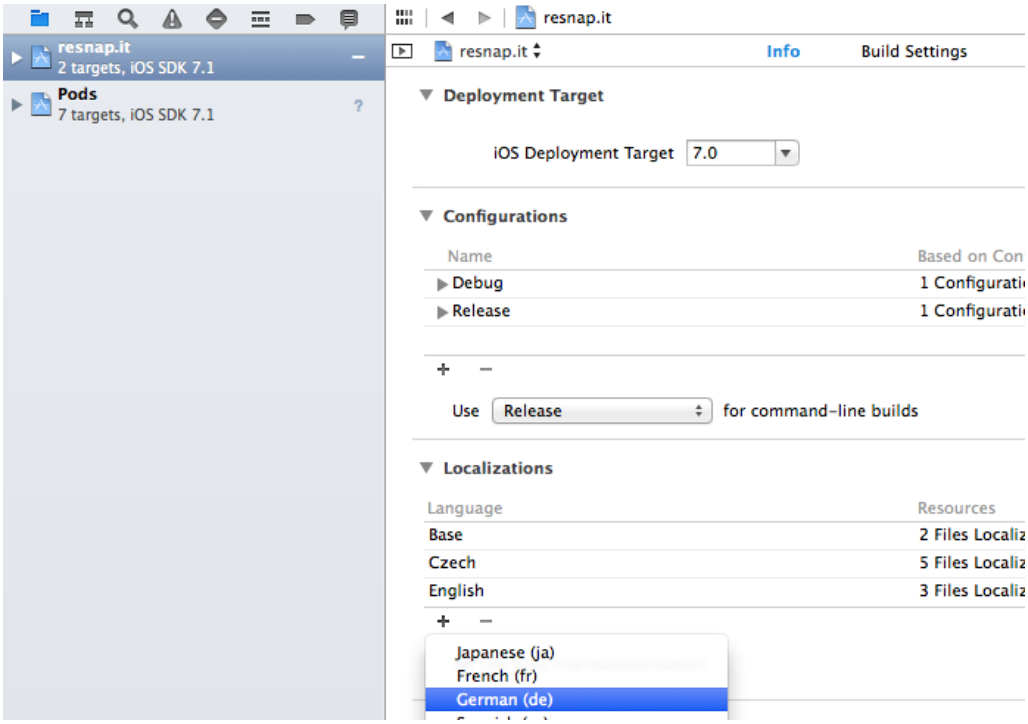
Výpis 12: Příklad použití knihovny Canvas

4.5 Lokalizace

Uživatelé mobilních zařízení pocházejí z různých zemí a mluví různými jazyky. Pro lepší uživatelský zážitek je důležité, aby aplikace podporovala vice jazyků. Proces překládání

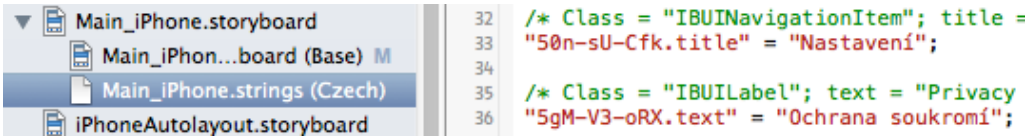
aplikace do místního jazyka se označuje jako lokalizace. U aplikací se nepřekládá pouze text, ale taky formát zobrazení numerických hodnot, dat a času. V iOS se aplikace standardně podřídí jazyku zařízení, na kterém jsou spuštěné, pokud je daná lokalizace v aplikaci dostupná. Nicméně není problém implementovat volbu jazyka přímo do aplikace. To se používá zejména u her.

Přidávání nové lokalizace se v *XCode* provádí v sekci *Project info*.



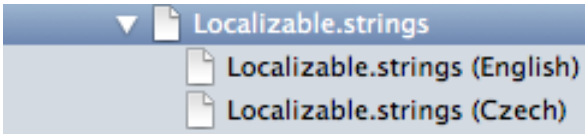
Obrázek 8: Přidávání lokalizace v *XCode*

XCode projede všechny textové prvky *storyboardu*, na základě kterých se vygeneruje *strings* soubor. Vybrané řetězce pak stačí přeložit.



Obrázek 9: Lokalizační soubor s již přeloženými řetězci v *XCode*

Některé řetězce se však generují dynamicky. Např. různé dialogy nebo parametrizované řetězce. Je třeba vytvořit nový *strings* soubor, a v sekci *File inspector* kliknout na *Localize*. Poté stačí zaškrtnout požadované jazyky a tím se vytvoří jednotlivé lokalizace.



Obrázek 10: *strings* soubor v *XCode*

Soubor se skládá z dvojic klíč–hodnota.

```
"CANCEL" = "Zrušit";
"OPEN_IN_BROWSER" = "Otevřít v prohlížečí";
"BASED_ON_%@_VOTES" = "hlasovalo: %@";
```

Obrázek 11: obsah *strings* souboru

V místě kde chceme použít dynamicky generované řetězce použijeme makro `NSLocalizedString`. Následující výpis kódu představuje metodu, která po kliknutí na dané tlačítko vyvolá menu obsahující 3 položky.

```
– (IBAction)showMoreActionSheet:(id)sender {
    UIActionSheet *actionSheet = [[UIActionSheet alloc]  initWithTitle  : nil
                                  delegate:self
                                  cancelButtonTitle :NSLocalizedString(@"CANCEL", nil)
                                   destructiveButtonTitle :NSLocalizedString(@"REPORT_SERIES",
                                       nil)
                                  otherButtonTitles :NSLocalizedString(@"OPEN_IN_BROWSER",
                                      nil), nil];
    [actionSheet showFromBarButtonItem:self.moreButton animated:YES];
}
```

Výpis 13: Příklad použití NSLocalizedString

4.6 Optimalizace pro 64-bit

Jednou z velkých novinek, kterou přinesl iPhone 5S představený 10. září 2013, je čip A7 postavený na 64-bitové architektuře. iPhone 5S se tak stal prvním smartphonem na světě, který používá 64-bitový procesor. iOS běžící na 64-bitovém zařízení obsahuje jak 32-bitové tak 64-bitové systémové knihovny. Pokud jsou všechny spuštěné aplikace v 64-bitové verzi, iOS nikdy nenačte 32-bitové verze knihoven[27]. Ve výsledku systém používá méně paměti a aplikace se načítají rychleji. Jelikož už všechny nativní aplikace podporují 64-bitovou architekturu, je velkým plusem, když jsou všechny aplikace běžící na 64-

bitovém zařízení také v 64-bitové verzi. Zvláště aplikace, které potřebují velký výpočetní výkon.

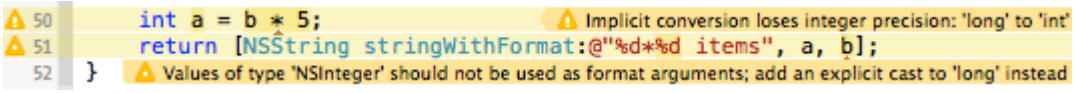
64-bitová architektura přinesla změny týkající se zejména datových typů. Např. `long` nebo `NSInteger` na 64-bitové architektuře mají daleko větší rozsah možných hodnot, než je tomu u 32-bitové architektury. Pokud tedy aplikace používá některý z datových typů, kterého se změna dotkla, výsledky výpočtu se mohou na 64-bitové architektuře lišit. Zejména pokud výsledek výpočtu převyšuje maximální hodnotu 32-bitové verze datového typu.

```
// 32-bitove prostredi
NSLog(@"LONG_MAX = %ld", LONG_MAX);
// vystup: LONG_MAX = 2147483647

// 64-bitove prostredi
NSLog(@"LONG_MAX = %ld", LONG_MAX);
// vystup: LONG_MAX = 9223372036854775807

// 32-bitove prostredi
int t = LONG_MAX;
NSLog(@"LONG_MAX = %ld", t);
// vystup: LONG_MAX = 2147483647

// 64-bitove prostredi
int t = LONG_MAX;
NSLog(@"LONG_MAX = %ld", t);
// vystup: LONG_MAX = 4294967295
```

Výpis 14: Porovnání datových typů na 32-bitové a 64-bitové architektuře

Pokud zkopírujeme data z většího datového typu do menšího datového typu, jejich hodnota může být ořezána. V 32-bitovém prostředí lze předpokládat, že dva určité datové typy jsou stejné, a je tedy bezpečné předávat hodnotu z jednoho typu do druhého. Tento předpoklad však nemusí být správný v 64-bitovém prostředí. Viz výpis 15: Porovnání datových typů na 32-bitové a 64-bitové architektuře.

Některé z těchto chyb je schopný najít *XCode* při kompilaci kódu. Pokud zkompilujeme aplikaci pro 64-bitové zařízení, *XCode* vypíše u problémových řádků varovná hlášení.



Obrázek 12: Varovná hlášení v *XCode*

Několik takovýchto problémů se objevilo i v aplikaci resnap.it. Jejich řešení však bylo poměrně triviální.

```
// puvodni metoda, vracejici cestu na serveru k obrazkum dane galerie
- (NSString*)createTimeSnapPathFromAuthorId:(NSInteger)timeSnapAuthorId
    TimeSnapAzureName:(NSString*)timeSnapAzureName {

    int userGroup = timeSnapAuthorId / 1000; //zde by mohlo na 64-bitovem prostredi dochazet k
        problemum
    return [NSString stringWithFormat:@"%d/author-%d/%@/", userGroup, timeSnapAuthorId,
        timeSnapAzureName]; //zde by mohlo na 64-bitovem prostredi dochazet k problemum
}

// metoda po optimalizaci pro 64-bitove zarizeni
- (NSString*)createTimeSnapPathFromAuthorId:(NSInteger)timeSnapAuthorId
    TimeSnapAzureName:(NSString*)timeSnapAzureName {

    NSInteger userGroup = timeSnapAuthorId / 1000;
    return [NSString stringWithFormat:@"%ld/author-%ld/%@/", (long)userGroup, (long)
        timeSnapAuthorId, timeSnapAzureName];
}
```

Výpis 15: Optimalizace metody pro 64-bitové zařízení

5 Distribuce

Pro distribuci aplikace se nejčastěji používá *App Store*. Každá aplikace nabízená skrze *App Store* musí splňovat řadu pravidel, která jsou popsané v dokumentaci pro vývojáře. Mezi tyto pravidla patří např.:

- Aplikace musí být stabilní
- Aplikace nesmí používat privátní API
- Pro všechny finanční transakce je nutné používat *In App Purchase*
- Aplikace nesmí používat kameru nebo mikrofón bez vědomosti uživatele

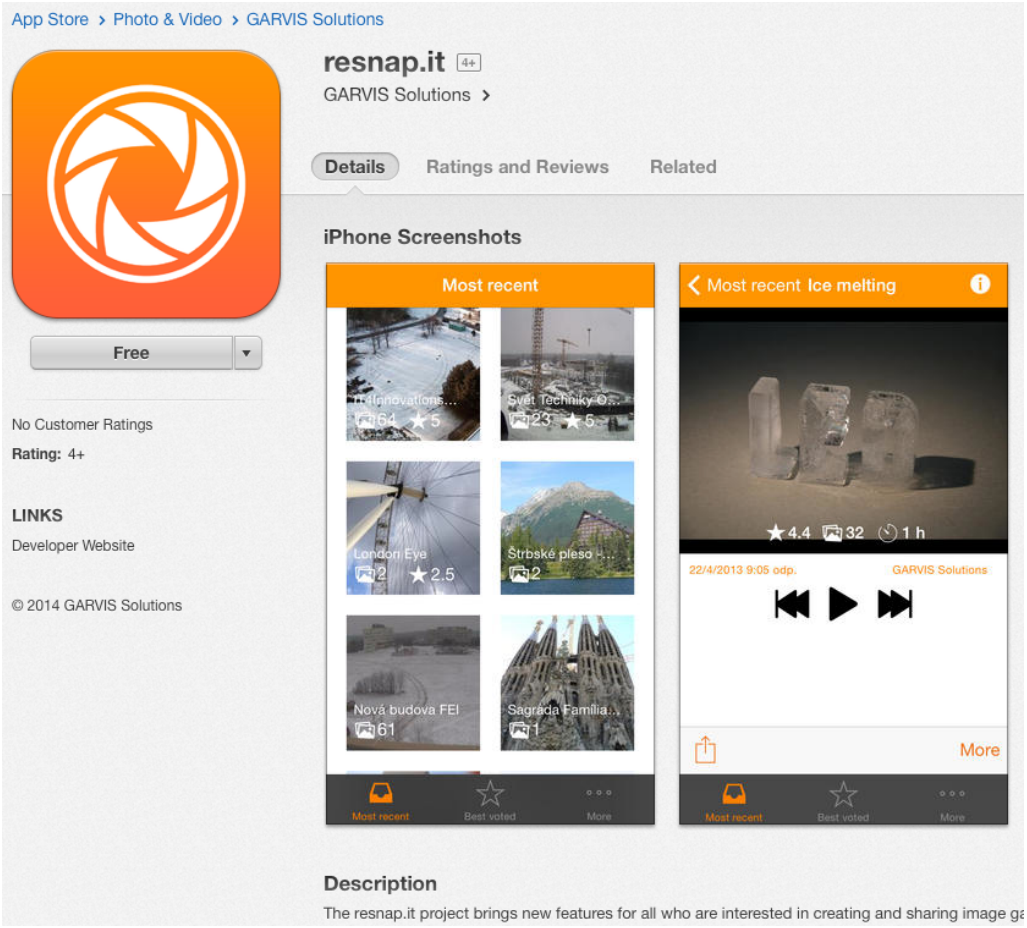
Před odesláním aplikace do *App Store* je třeba mít *App ID*, distribuční certifikát a *Provisioning Profile*. Všechno se provádí pomocí nástrojů v *XCode*. Dále je třeba mít ikony, viz následující tabulka[20]:

Popis	iPhone a iPod Touch (vysoké rozlišení)	iPad a iPad mini (vysoké rozlišení)	iPad a iPad mini (standardní rozlišení)
Ikona aplikace	120 x 120	152 x 152	76 x 76
Ikona pro *App Store*	1024 x 1024	1024 x 1024	1024 x 1024
Ikona pro vyhledávání	80 x 80	80 x 80	40 x 40
Ikona nastavení	58 x 58	58 x 58	29 x 29

Tabulka 2: Velikost ikon v pixelech

Samozřejmě, pokud je aplikace pouze pro iPhone/iPod Touch, není nutné mít ikony pro iPad/iPad mini, a naopak. Doporučuje se zvolit formát PNG. Rovněž je třeba připravit snímky obrazovky aplikace. Každá aplikace může mít až pět snímků, povinný je alespoň jeden snímek. V případě že se jedná o univerzální aplikaci, je nutné dodat snímky jak pro iPhone, tak pro iPad verzi. Snímky aplikace jsou důležité a je to často jediný faktor, podle kterého se uživatel rozhodne, zda si aplikaci stáhne, či nikoliv.

Na konec je třeba připravit metadata. Jedná se o název aplikace, číslo verze, primární (a nepovinná sekundární) kategorie, popis aplikace, klíčová slova a URL adresa s podporou aplikace[28]. Pokud se jedná o novou verzi aplikace, lze rovněž vyplnit pole „Co je nového v této verzi“.



Obrázek 13: Výsledná aplikace v *App Store*

Následně je třeba přihlásit se do webového portálu *iTunes Connect*, kde se vytvoří nová aplikace, vyplní základní informace, cena, dostupnost a metadata. Poté je třeba vytvořit archiv samotné aplikace, který se nahraje do *App Store*. To se provádí v *XCode* a je nutné mít k počítači připojené reálné iOS zařízení. Pokud vše proběhlo v pořádku, status aplikace se změní na „Waiting for Review“. Apple každou aplikaci před vydáním do *App Store* kontroluje, a pokud by např. aplikace porušovala výše zmíněná pravidla, vrátil by ji k opravě. Tento proces trvá zpravidla několik dní.

6 Závěr

Předmětem této práce bylo analyzovat operační systém iOS, zejména z pohledu vývoje pro tuto platformu. V první kapitole byly popsány její vlastnosti a specifika, a také jazyk Objective-C, používaný k vývoji mobilních aplikací na dané platformě. V druhé kapitole byl rozebrán proces návrhu mobilní aplikace. Uživatelské rozhraní aplikace je u mobilních zařízení velmi důležité. Často určuje, zda se bude uživatel k aplikaci vracet, či nikoliv. Čtenář byl seznámen s jeho klíčovými prvky a s jeho tvorbou.

Dalším cílem bylo vyvinout mobilní aplikaci. Výsledkem je aplikace resnap.it, která je volně ke stažení v *App Store*. Ikdyž se jedná o prohlížeč časosběrných snímků, již teď jsou naimplementovány další funkce, které souvisí např. s pořizováním snímků. Distribuovat vlastnosti, které nejsou plně funkční, nemá smysl. Práce na vývoji aplikace tedy není u konce, a v budoucnu lze očekávat další verze s rozšířenou funkcionalitou. Vybrané klíčové prvky její implementace byly popsány ve třetí kapitole. Byl zde také popsán efektivní způsob používaní knihoven třetích stran.

Nasazení hotové aplikace do *App Store* je poměrně složitý proces, vyžadující delší přípravu. Tento proces je stručně popsán v poslední kapitole.

Filozofie OS X se mi velmi zamlouvá, jako vývojář zejména oceňuji příkazovou řádku. *XCode* je mocným nástrojem, a každá verze přináší mnoho vylepšení. Spolu s *iOS Simulator* umožňuje velmi rychlé a pohodlné testování. Objective-C se velmi liší od ostatních jazyků. Chvíli tedy trvá, než si na něj člověk zvykne. Dokumentace je však velmi dobře napsaná a na internetu existuje velká komunita lidí, kteří na toto téma píšou články, či přispívají do *open-source* knihoven třetích stran. Samotný vývoj pro iOS vnímám tedy velmi pozitivně a této problematice se určitě budu věnovat i v budoucnu.

Adam Bardoň

7 Reference

[1] P. Buttfield-Addison, *Learning Cocoa with Objective-C: Developing for the Mac and iOS App Stores*, 2012, O'Reilly Media, ISBN: 978-1449318499.

[2] J. Conway, *iOS Programming: The Big Nerd Ranch Guide*, 2013, Big Nerd Ranch Guides, ISBN: 978-0321942050

[3] Kochan, Stephen G., *Programming in Objective-C* (4th ed.), 2011, Addison-Wesley, ISBN: 978-0321811905

[4] M. Neuburg, *iOS 7 Programming Fundamentals: Objective-C, Xcode, and Cocoa Basics*, 2013, O'Reilly Media, ISBN: 978-1491945575

[5] C. Banga; J. Weinhold, *Essential Mobile Interaction Design: Perfecting Interface Design in Mobile Apps (Game Design/Usability)*, 2014, Addison Wesley, ISBN: 978-0321961570

[6] FOX, Zoe. iPads Have Crippled the Growth of PC Sales. In: *Mashable.com* [online]. 2013-10-10 [cit. 2014-03-05]. Dostupné z: `http://mashable.com/2013/10/10/ipad-cripples-pc-growth/`

[7] KROES, Neelie. The €63 billion app boom. Nearly 5 million jobs in European app sector by 2018, says EU report. In: *Europa.eu* [online]. 2014-02-13 [cit. 2014-04-29]. Dostupné z: `http://europa.eu/rapid/press-release_IP-14-145_en.htm`

[8] RIVERA, Janessa. Gartner Says Mobile App Stores Will See Annual Downloads Reach 102 Billion in 2013. In: *Gartner.com* [online]. 2013-09-19 [cit. 2014-04-29]. Dostupné z: `http://www.gartner.com/newsroom/id/2592315`

[9] LUNDEN, Ingrid. Gartner: 102B App Store Downloads Globally In 2013, $26B In Sales, 17% From In-App Purchases. In: *Techcrunch.com* [online]. 2013-09-19 [cit. 2014-04-29]. Dostupné z: `http://techcrunch.com/2013/09/19/gartner-102b-app-store-downloads-globally-in-2013-26b-in-sales-17-from-in-app-purchases/`

[10] Apple. *Wikipedia: the free encyclopedia*. [online]. 2001- [cit. 2014-03-15]. Dostupné z: `http://cs.wikipedia.org/wiki/Apple`

[11] Startup. *Wikipedia: the free encyclopedia*. [online]. 2001- [cit. 2014-03-15]. Dostupné z: `http://cs.wikipedia.org/wiki/Startup`

[12] GURMAN, Mark. Apple's iTunes Radio launches internationally, starting with Australia. In: *9to5mac.com* [online]. 2014-02-10 [cit. 2014-03-16]. Dostupné z: http://9to5mac.com/2014/02/10/apples-itunes-radio-launches-internationally-starting-with-australia/

[13] INGRAHAM, Nathan. Apple announces 1 million apps in the App Store, more than 1 billion songs played on iTunes radio. In: *Theverge.com* [online]. 2013-10-22 [cit. 2014-03-16]. Dostupné z: http://www.theverge.com/2013/10/22/4866302/apple-announces-1-million-apps-in-the-app-store/

[14] OSx86. *Wikipedia: the free encyclopedia*. [online]. 2001- [cit. 2014-03-16]. Dostupné z: http://en.wikipedia.org/wiki/Hackintosh

[15] Objective-C. *Wikipedia: the free encyclopedia*. [online]. 2001- [cit. 2014-03-18]. Dostupné z: http://en.wikipedia.org/wiki/Objective-C

[16] Blocks (C language extension). *Wikipedia: the free encyclopedia*. [online]. 2001- [cit. 2014-03-18]. Dostupné z: http://en.wikipedia.org/wiki/Blocks_(C_language_extension)

[17] Start Developing iOS Apps Today, *Apple Inc.*, [online]. 2013-10-22 [cit. 2014-03-18]. Dostupné z: https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/RoadMapiOS.pdf

[18] App Store Distribution. *Apple Inc.* [online]. 2014 [cit. 2014-03-19]. Dostupné z: https://developer.apple.com/support/appstore/

[19] iOS App Programming Guide, *Apple Inc.*, [online]. 2013-10-23 [cit. 2014-03-20]. Dostupné z: https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/iPhoneAppProgrammingGuide.pdf

[20] iOS Human Interface Guidelines, *Apple Inc.*, [online]. 2014-03-10 [cit. 2014-03-20]. Dostupné z: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf

[21] DOWA, Johann. Apple announces 1 million apps in the App Store, more than 1 billion songs played on iTunes radio. In: *theverge.com* [online]. [cit. 2014-03-23]. Dostupné z: http://maniacdev.com/ios-5-sdk-tutorial-and-guide/xcode-4-storyboard/

[22] Cocoa Application Competencies for iOS: Storyboard, *Apple Inc.*, [online]. 2013-09-18 [cit. 2014-03-25]. Dostupné z: https://developer.apple.com/library/IOS/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html

[23] *Cocoapods* [online]. [cit. 2014-03-27]. Dostupné z: http://cocoapods.org

[24] POITREY, Olivier. SDWebImage on GitHub, [online] [cit. 2013-03-28] Dostupné z: https://github.com/rs/SDWebImage

[25] CREUZOT, Alexis. KASlideShow on GitHub, [online] [cit. 2013-03-28] Dostupné z: https://github.com/kirualex/KASlideShow

[26] TANG, James. Canvas on GitHub, [online] [cit. 2013-03-29] Dostupné z: https://github.com/CanvasPod/Canvas

[27] 64-Bit Transition Guide for Cocoa Touch, *Apple Inc.*, [online]. 2014-02-11 [cit. 2014-03-29]. Dostupné z: https://developer.apple.com/library/ios/documentation/General/Conceptual/CocoaTouch64BitGuide/CocoaTouch64BitGuide.pdf

[28] JACOBS, Bart. How To Submit an iOS App to the App Store. In: *Tutsplus.com* [online]. 2013-05-06 [cit. 2014-04-10]. Dostupné z: http://code.tutsplus.com/tutorials/how-to-submit-an-ios-app-to-the-app-store--mobile-16812/

A Příloha na CD

Obsah CD je organizován do následujících adresářů:

- animace - obsahuje obsahuje dvě animace výsledné aplikace ve formátech .gif a .mov
- resnap.it - obsahuje zdrojový kód výsledné aplikace, který lze prohlížet v libovolném textovém editoru, ideálně však v programu *XCode*